University of Massachusetts: MUC-4 Test Results and Analysis

W. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, & S. Soderland

Department of Computer Science
University of Massachusetts
Amherst, MA 01003
lehnert@cs.umass.edu

INTRODUCTION

The UMass/MUC-4 system is based on a form of sentence analysis known as ***selective concept extraction***. This approach to language processing is distinguished by a minimal reliance on syntactic sentence analysis, along with a minimal dictionary customized to operate in a limited domain. Last year, the UMass/MUC-3 system demonstrated the viability of selective concept extraction, but serious questions were raised about the portability and scalability of the technology, particularly with respect to the creation of domain-dependent and task-dependent dictionaries. We estimated that 9 person/months went into the creation of the dictionary used by UMass/MUC-3, and we were unable to say how much domain-dependent lexicon was still missing. We were nevertheless sure that our dictionary coverage was incomplete.

This year we confronted the issue of efficient system development, with particular attention to the problem of dictionary construction. As a result, we are now in a position to claim that effective customized dictionaries can be constructed quickly and easily by relatively inexperienced system developers. The dictionary used by UMass/MUC-3 emerged after roughly 1500 hours of highly skilled labor by two advanced graduate students and one post doc. For MUC-4, we created a new dictionary that achieved nearly the full functionality of the UMass/MUC-3 dictionary after only 8 hours of effort on the part of a first-year graduate student. This outcome was achieved through the use of an automated dictionary construction tool called AutoSlog. The AutoSlog dictionary was used for our optional TST3 and TST4 runs, while a modified version of our UMass/MUC-3 dictionary was used for our official TST3 and TST4 runs. Our optional and official systems were identically configured except for their dictionaries.

Our official UMass/MUC-4 system employs a new memory-based consolidation module for discourse analysis, some generic enhancements to the CIRCUS sentence analyzer, and filters that operate after sentence analysis and then again after consolidation in order to reduce spurious slot fills and spurious templates. We also made a number of adjustments associated with MUC-4 template updates and domain specifications. We found it necessary to eliminate last year's optional case-based consolidation module because of its strong tendency toward overgeneration. Even so, we had two competing consolidation modules to evaluate, as well as a variety of possible filters. To resolve all of these other system options, we ran hundreds of tests over TST1, TST2 and 250 additional texts from the development corpus.

OFFICIAL TESTING AND RESULTS

We ran four test sets for MUC-4. Our official system was run on TST3 and TST4 as required, and we ran one additional optional system on TST3 and TST4. The official system and optional system were identical except for their dictionaries. The optional system ran a dictionary constructed by AutoSlog that contained 379 concept node definitions. Our official system ran with a version of the UMass/MUC-3 dictionary augmented by 76 additional concept node definitions imported from the AutoSlog dictionary (for a total of 389 concept node definitions). Both dictionaries accessed the same 5436 lexical definitions for part-of-speech recognition (these definitions were taken from the UMass/MUC-3 dictionary), along with 2102 proper names. We predicted that both systems would produce comparable levels of precision, and that the optional system would fall behind the official system by 10 recall points under All Templates. Table 1 contains the All Templates scores for all four test runs.

Our official and optional systems both crashed twice on TST3, once on a relevant text containing three key templates and once again on an irrelevant text. No system crashes occurred during TST4. We note that the official system was run on all of DEV, TST1, and TST2 without any fatal errors shortly before the official testing began.

Report Documentation Page	*Form Approved OMB No. 0704-0188*

1. REPORT DATE **1992**	2. REPORT TYPE	3. DATES COVERED **00-00-1992 to 00-00-1992**
4. TITLE AND SUBTITLE **University of Massachusetts: MUC-4 Test Results and Analysis**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Massachusetts,Center for Intelligent Information Retrieval,Department of Computer Science,Amherst,MA,01003**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**8**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

As predicted, the AutoSlog dictionary produced lower recall levels than the official system: 7 points lower for TST3, and 12 points lower for TST4. Precision was comparable for both systems with AutoSlog generating higher overall precision rates: 2 points higher for TST3 and 1 point higher for TST4. We note that our performance on TST4 was generally worse than TST3. However, a close inspection of the detailed score reports for the official system shows that the primary difference in those reports lies in the All Templates precision scores: 57 for TST3 vs. 45 for TST4. This loss of precision can be explained for the most part by comparing the number of spurious templates: 16 for TST3 vs. 31 for TST4.

System	recall	precision	P&R	2P&R	P&2R
Official TST3	47	57	51.52	54.67	48.71
Optional TST3	40	59	47.67	53.88	42.75
Official TST4	48	45	46.45	45.57	47.37
Optional TST4	36	46	40.39	43.58	37.64

Table 1: Overall Scores under All Templates for the Four UMass/MUC-4 Test Runs

Setting aside the differences between TST3 and TST4, we were pleased to see how well the AutoSlog dictionary performed relative to our hand-crafted dictionary from last year. Comparing P&R scores, our AutoSlog dictionary achieved 93% of the overall performance of our official system on TST3, and 87% of the official system's performance on TST4. In an effort to leverage the UMass/MUC-3 and the AutoSlog dictionaries, we strengthened the performance of the MUC-3 dictionary by augmenting it with 76 AutoSlog definitions.[1] Without this boost to the MUC-3 dictionary, the distance between our official and optional systems would have been insignificant.

Given our MUC-4 test results, we have demonstrated that an effective domain-dependent dictionary can be efficiently constructed using a representative text corpus accompanied by hand-coded template encodings. Our preliminary and very limited efforts have produced a dictionary that closely mirrors the functionality obtained by a relatively successful hand-crafted dictionary. Although the process of dictionary construction via AutoSlog is not totally automated, the manual labor needed can be completed in a matter of hours by a single individual with minimal expertise in dictionary construction. We consider this to be a significant step forward in the area of automated dictionary construction for text extraction applications.

AUTOMATED DICTIONARY CONSTRUCTION

The AutoSlog construction tool analyzes available key templates in conjunction with source texts and generates hypothesized CIRCUS definitions without human assistance. AutoSlog's proposed concept node definitions are derived from sentences in the MUC-4 development corpus that contain string fills associated with key templates. Using the complete 1300-text DEV corpus as the training set for our dictionary construction experiment, AutoSlog proposed 1356 concept node definitions in response to 1272 string-fill slots. Although a large number of these definitions were flawed or redundant, 28% of AutoSlog's proposed definitions were reasonable and could be included in an operational dictionary without alteration. In our experiment, 375 of the 1356 definitions proposed by AutoSlog were deemed acceptable when reviewed by visual inspection.

Each AutoSlog definition begins with a single string fill in a single key template. Given a specific slot, AutoSlog extracts the first non-empty string fill listed in the key template (string-fill slots often contain multiple strings based on multiple references within the source text). It then searches the source text for the first instance of that exact string within the source text. Once located, AutoSlog pulls the complete sentence containing that string from the source text and passes it to the CIRCUS sentence analyzer for syntactic analysis. CIRCUS analyzes the sentence using a part-of-speech dictionary. When all goes well, a set of buffers are instantiated with simple syntactic constituents corresponding to a subject, a verb, and possibly an object or a prepositional phrase. When the original string fill shows up in one of these buffers, AutoSlog hypothesizes a concept node definition complete with a lexical trigger, complement pattern, and slot constraints. This definition is then written to a file and AutoSlog returns to the key template for the next string fill slot. Figure 1 shows the AutoSlog construction tool in action.

[1] Other methods of leveraging the two dictionaries were tested, but this was the most effective strategy.

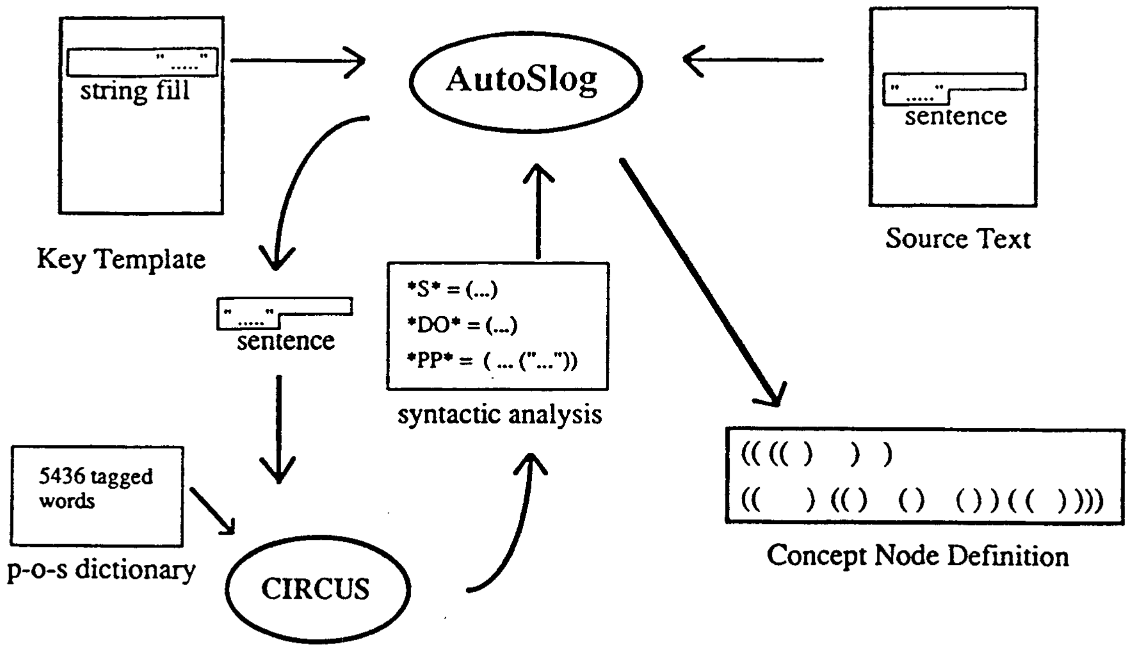

Figure 1: Automated Dictionary Construction

The presence of string-fills in key templates is a crucial requirement for AutoSlog. In fact, the more string-fill slots, the better. The MUC-4 templates contained six string-fill slots used by AutoSlog: inc-instr-id, perp-ind-id, perp-org-id, phys-tgt-id, hum-tgt-name, and hum-tgt-desc. After processing the 1300 texts of DEV, AutoSlog generated 136 definitions based on inc-instr-id, 316 definitions from perp-ind-id, 201 definitions from perp-org-id, 306 definitions from phys-tgt-id, 193 definitions from hum-tgt-name, and 204 definitions from hum-tgt-desc. This dictionary was compiled in 14 hours and then passed to a CIRCUS programmer for manual review. During the review process, each definition was dispatched into one of two possible states: (1) keep as is, or (2) save for possible revision. Files were maintained for the "keeps" and the "edits" with the expectation that the keep-definitions might be augmented by some number of edit-definitions if any of the edit definitions could be salvaged. The initial categorization into "keeps" and "edits" was relatively fast because each definition could be categorized on the basis of visual inspection alone. Many definitions destined for the edit files were easy to spot since they often resulted from parsing errors, patterns of no linguistic generality, or patterns of dubious reliability.

Here is an example a good AutoSlog definition generated by the first text in the development corpus:

```
Id: DEV-MUC3-0001        Trigger: KIDNAPPED        Trigger Root: KIDNAP        Syntactic-type: VERB
Slot filler: "TERRORISTS"
Sentence: (THE ARCE BATTALION COMMAND HAS REPORTED THAT ABOUT &&50 PEASANTS OF VARIOUS AGES HAVE
        BEEN KIDNAPPED BY TERRORISTS OF THE FARABUNDO_MARTI_NATIONAL_LIBERATION_FRONT IN
        SAN_MIGUEL DEPARTMENT >PE)
Name: %ACTOR-PASSIVE-VERB-PP-KIDNAPPED-BY%
Time limit: 10
Variable Slots:                      Constraints:
(ACTOR (*PP* (IS-PREP? '(BY))))      (((CLASS ORGANIZATION *PP*)
                                      (CLASS TERRORIST *PP*)
                                      (CLASS PROPER-NAME *PP*)
                                      (CLASS HUMAN *PP)))
Constant Slots: (TYPE PERPETRATOR)
Enabling Conditions: ((PASSIVE))
```

This definition extracts slot fillers from constructions of the form: "X is/was/(has/have been) kidnapped by Y." This particular definition will only pick up the conceptual actor Y. A separate definition is needed to pick up the conceptual victim X. The following is an example of a bad AutoSlog definition:

Id: DEV-MUC3-0036 Trigger: WAS Trigger Root: WAS Syntactic-type: VERB
Slot Filler: "MEMBER OF THE DEMOCRATIC SOCIALIST PARTY"
Sentence: (GILDA FLORES WAS AN ACTIVE MEMBER OF THE DEMOCRATIC SOCIALIST PARTY OF GUATEMALA >CO WHOSE SECRETARY_GENERAL MARIO SOLORZANO REPORTED SALVADORAN PARAMILITARY GROUPS ARE CARRYING_OUT ACTIONS IN THIS COUNTRY >PE)
Name: %VICTIM-ACTIVE-OBJECT-VERB-WAS%
Time Limit: 10

Variable Slots:	Constraints:
(VICTIM (*DO* 1))	(((CLASS HUMAN *DO*)
	(CLASS PROPER-NAME *DO*)))

Constant Slots: (TYPE KIDNAPPING)
Enabling Conditions: ((ACTIVE :CHECK-DO-NO-ONE T))

As it stands, this definition hypothesizes that an active form of the verb "to be" predicts the victim of a kidnapping. Although the source sentence does legitimately suggest that the verb "to be" can be used to link human names with human descriptions, this proposed definition cannot be trusted to deliver a kidnapping victim.

When AutoSlog creates a new definition, it checks the existing set of previously proposed definitions to see if the current proposal duplicates an older one. AutoSlog does not produce multiple copies of the same definition. By tracking the number of duplicates AutoSlog suppresses, we can see evidence that the dictionary is approaching a saturation point. In particular, we note that after AutoSlog has processed 1200 texts, the next 100 texts generate only half as many definitions as the first 100 texts. Figure 2 shows the weakening frequency of new dictionary definitions as we move through the development corpus.

Although the AutoSlog dictionary definitions are derived from only six template slots, consolidation and template-generation routines are capable of extracting the information needed to fill additional slots. When the AutoSlog dictionary operates in conjunction with the full system, we can fill every template slot except phys-tgt-nation, phys-tgt-effect, phys-tgt-total-num, and hum-tgt-total-num.

The 8-hour AutoSlog dictionary was completed only four weeks before the final testing for MUC-4. Now that we have seen how much impact AutoSlog can have on the process of dictionary construction, it makes sense to pursue enhancements to AutoSlog in order to strengthen its baseline performance. As it stands, AutoSlog can be moved to new domains with a minimal amount of software tuning. Adjustments must be made to handle a new template design, but any templates that contain string-fills will serve to fuel dictionary construction.

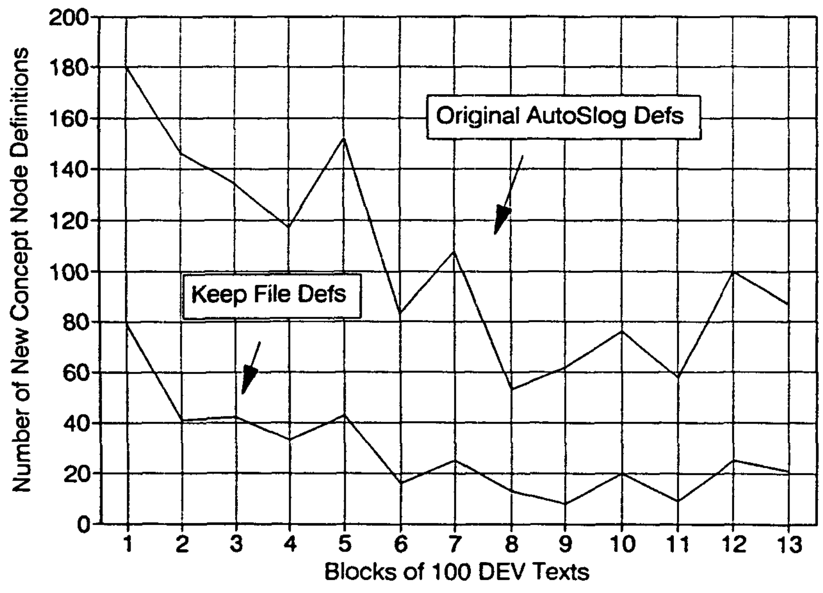

Figure 2: Dictionary Saturation Under AutoSlog

TST3 ERROR ANALYSIS

We have conducted a post hoc analysis of our system's performance on TST3 in order to better understand the various problems encountered on TST3. Most of this data describes the behavior of CIRCUS, its use of concept node definitions, and the effects of memory-based consolidation. As detailed and useful as the score reports are, score reports are not designed to tease apart the performance contributions of a sentence analyzer, discourse analyzer, or template generator. Subcomponents like these must be analyzed separately if we want to understand where to focus future development efforts.

Recall Limitations

The dictionary used by the official UMass/MUC-4 system contained 389 concept node definitions. Of these, 172 (44%) were enabled[2] to process TST3. On average, each definition was enabled nearly 9 times for a total of 1515 concept node enablements (~15 per text on average). For TST3, CIRCUS extracted 943 string fills to fill variable slots in enabled concept nodes. Because there are a lot of redundant concept node definitions, almost half of these string fills were duplicates, leaving 520 unique string fills extracted by CIRCUS. According to our analysis, 214 of these string fills were discarded during consolidation and 306 string fills made it into a response template.

Of the 520 non-redundant string-fills, 38% were correctly incorporated into a response template where they matched the string or strings listed in a key template. A full 34% were correctly discarded or merged by consolidation (and therefore did not make it into a response template). The sum of these instances accounts for 72% of the total string fills - all handled correctly by consolidation. Of the remaining string fills, 21% appeared in response templates as spurious slot fills, and 7% were incorrectly discarded. Of the 237 string fills that did legitimately correspond to slot fills in key templates, consolidation correctly incorporated 199 (84%) into response templates. Even so, our overall recall score was only 46%. Where are the other string fills?

Our analysis shows that CIRCUS generated 225 good (full match) string fills and 12 partially good (partial match) string fills. According to the score report, there were 416 possible string fills for TST3. That tells us CIRCUS is producing only 55% of the possible string fills for TST3. This 55% hit rate effectively imposes a rough ceiling on our overall recall, and suggests that significant gains in recall will require stronger performance levels during sentence analysis.

PRECISION LIMITATIONS

When we examine the 306 string fills present in our TST3 response templates, we find that 187 could be matched to slot fills in some key template and 12 could be partially matched to a key template slot fill. If all of these string fills were in correct slots and correct templates, our string fill precision would be 63%. But only 142 string fills result in full matches with key template slots and 24 result in partial matches. Of the 107 strings that can't be matched to any key template, we have found the following breakdown of errors:

49 (46%) should have been discarded as irrelevant
16 (15%) were from mis-fired concept node definitions
15 (14%) were from parser errors
14 (13%) should have been merged with a more specific string
12 (11%) were from words not covered adequately by the dictionary
1 (1%) was from a source string altered by preprocessing

[2] An enabled concept node is one that produces a case frame instantiation. All output generated by CIRCUS is based on enabled concept nodes.

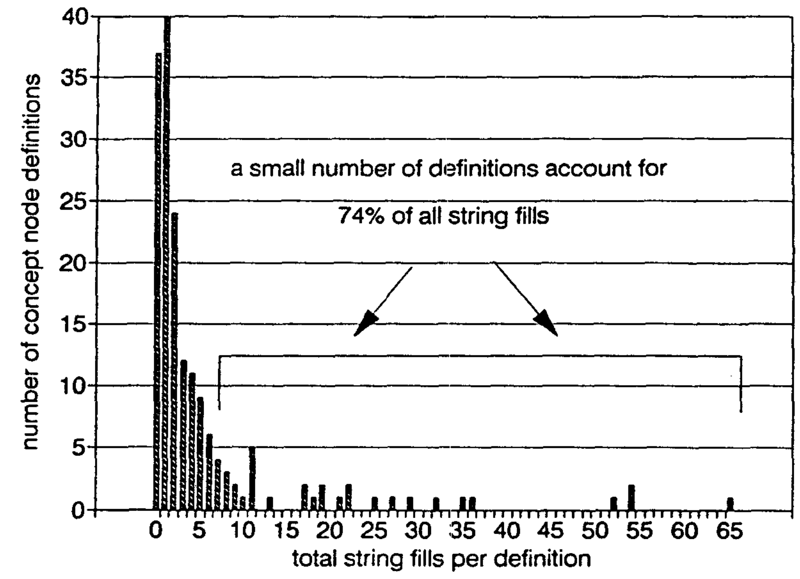

Figure 3: Concept Node Frequency Distribution

Of the 49 false hits associated with relevancy discriminations, our single greatest precision error came from 30 false hits on military clashes. After that, four general problem areas were about equally responsible for a significant number of errors: (1) false hits associated with faulty concept node definitions, (2) CIRCUS sentence analysis errors, (3) consolidation merging failures, and (4) inadequate dictionary coverage.

Dictionary Coverage

Although inadequate dictionary coverage was identified as a source of visible precision loss, we have been remarkably well-served by a relatively small dictionary of 5436 lexical items augmented by 2102 proper names. An analysis of the TST3 lexicon shows that 1008 words appearing in TST3 were not recognized by our system. Of these, 696 occurred only once. Of the remaining 312 words, the vast majority were proper names. A visual inspection of the unrecognized word list suggested that our lexicon was apparently adequate for the demands of TST3. However, this does not mean that all of our associated definitions were above reproach.

Although our dictionary contains a total of 389 concept node definitions, only 172 of these were used during TST3. A frequency analysis of these 172 definitions showed that 20% of the definitions generated 74% of the string fills. A total of 37 (22%) concept node definitions failed to produce any string fills (perhaps these contain no variable slots or maybe they were discarded during consolidation), while one concept node definition produced 65 string fills, and three others produced over 50 string fills (each). Figure 3 shows the complete frequency distribution.

Comparing TST3 and TST4

Although our F-scores suggest dramatic differences between TST3 and TST4, there appears to be a single factor that is responsible for these divergent score summaries. Table 2 shows a more detailed perspective on the differences and similarities between TST3 and TST4.

Reviewing the scores in Table 5, we see that there is a remarkable similarity across most of the scores for TST3 and TST4. TST4 was even better than TST3 under String Fills Only. The major differences appear in the precision and overgeneration scores for Matched/Spurious and All Templates. These differences also correspond to a striking difference in the number of spurious templates generated for TST3 (16) and TST4 (31). Looking deeper into the problem, we determined that two factors seemed to contribute to the large number of spurious templates in TST4.

	TST3			TST4		
	REC	PRE	OVG	REC	PRE	OVG
MATCHED/MISSING	47	67	14	48	69	13
MATCHED/SPURIOUS	60	57	26	63	45	43
MATCHED/ONLY	60	67	14	63	69	13
ALL TEMPLATES	47	57	26	48	45	43
SET FILLS ONLY	50	71	13	51	74	13
STRING FILLS ONLY	37	57	20	43	64	15
F-SCORES	51.52	54.67	48.71	46.45	45.57	47.37

Table 2: TST3 and TST4 score reports from the official UMass/MUC-4 test runs

First, many legitimate templates were deemed spurious because of mapping problems. We can see some evidence of this by running a comparative test designed to assess the impact of templates lost due to incorrect incident-type slot fills. In the comparative test, we will use "ATTACK" as the slot fill for all incident-type slots. This will ensure that no template is deemed spurious because an incident-type is blocking it from being mapped to a key template. Table 3 shows the resulting F scores from comparative test runs for both TST3 and TST4, running the official UMass/MUC-4 system and generating batch score reports throughout.

	P&R	2P&R	P&2R
TST3 - official system	46.47	49.64	43.68
TST3 - "all attacks" is on	45.46	48.63	42.67
TST4 - official system	39.44	38.56	40.36
TST4 - "all attacks" is on	40.98	40.38	41.58

Table 3: The Effect of Spurious Templates Lost to Incorrect Incident-Types

In comparing the net effect of the "all attacks" heuristic, we find that there is no advantage to any of the F-scores when all templates are typed as attacks in TST3. Indeed, there is a uniform drop of one point across the board when "all attacks" is turned on. On the other hand, the F scores for TST4 all benefit from the "all attacks" heuristic. P&R goes up 1.54, 2P&R goes up 1.82, and P&2R goes up 1.22. This tells us that we did not tend to lose otherwise legitimate templates because of their incident types in TST3, whereas a significant number of legitimate templates were lost for this reason in TST4. Precision is most dramatically affected by these errors, but P&R may have lost at least 2 points because of this problem (it is not possible to extrapolate from batch scores to interactive scores with complete certainty).

Second, a large number of spurious templates were created when military targets were not recognized to be military in nature. We have already seen how military clashes were the single greatest source of spurious string fills in TST3. Even so, we generated only 3 spurious templates due to false hits on military targets in TST3. In TST4 we generated 12 spurious templates for the same reason. If we assume that each spurious template contains 11 slot fills (a reasonable assumption when template filtering is in place), it follows that 132 spurious slot fills are coming from false hits on military targets in TST4. Removing 132 spurious slot fills from the TST4 score report, the precision score under All Templates goes from 42 to 48, and the P&R score goes up about 3 points as a result.

RESOURCES, SPIN-OFFS, AND FUTURE RESEARCH

Our primary system development and testing took place on three Texas Instruments Explorer II workstations each configured with 8 megabytes of RAM. Two Texas Instrument MicroExplorers, each with 8 megabytes of RAM, were used for the development of the AutoSlog lexicon. All development was done in Common Lisp. The system code and lexicons required 14 megabytes of storage on a Vax VMS fileserver, 8 of which comprised the

AutoSlog development files. System output was stored on a Decstation running Ultrix, where the scoring program was run. System testing required 250 megabytes of storage for response files and score reports. The two official test sets, TST3 and TST4, each took about 75 minutes to process on an Explorer II, using the workstation's local disk for all file output.

The development of our UMass/MUC-4 system has continued off and on for two years now. Last year we estimated that 2.25 person/years was invested in the UMass/MUC-3 system. This year we were able to build on that investment, and focus our effort more effectively on the critical issues of portability and scalability. All told, we estimate that one person/year of effort went into MUC-4 after MUC-3: 30% on testing, maintenance, and data analysis; 30% on CIRCUS enhancements and dictionaries; 15% on memory-based consolidation; 15% on MUC-4 updates and documentation; 10% on training for new personnel.

In the period since MUC-3, a number of related research projects have been pursued that did not directly contribute to MUC-4, but which address basic research issues associated with CIRCUS and text extraction systems. We have demonstrated that case-base reasoning and machine learning techniques can be successfully applied to the disambiguation of relative pronouns [1, 2, 3]; experiments have shown how CIRCUS can be used to support relevancy feedback algorithms for text classification [5]; and additional experiments have been conducted with a stochastic database derived from the MUC-3 corpus [4].

We expect to see similarly successful spin-offs from MUC-4 in the areas of automated dictionary construction and automated system scale-up. We will continue to exploit the MUC-3 corpus in pursuing these new directions, and we expect to gain additional experience with at least one new application domain as well.

ACKNOWLEDGEMENTS

This work was supported in part by the Defense Advance Research Projects Agency and monitored by Science Applications International Corporation under contract No. N666001-90-D-0192; DO#22.

BIBLIOGRAPHY

[1] Cardie, C. (1992a) "Corpus-Based Acquisition of Relative Pronoun Disambiguation Heuristics" to appear in *Proceedings of the 30th Annual Conference of the Association of Computational Linguisitcs.* University of Delaware, Newark DE.

[2] Cardie, C. (1992b) "Learning to Disambiguate Relative Pronouns" to appear in *Proceedings of the Tenth National Conference on Artificial Intelligence.* San Jose, CA.

[3] Cardie, C. (1992c) "Using Cognitive Biases to Guide Feature Set Selection" to appear in *Proceedings, Fourteenth Annual Conference of the Cognitive Science Society,* University of Indiana, Bloomington, IA.

[4] Fisher, D. and Riloff, E. (1992) "Applying Statistical Methods to Small Corpora: Benefiting from a Limited Domain" to appear in *Probabilistic Approaches to Natural Language,* a AAAI Fall Symposium. Cambridge, MA.

[5] Riloff, E. and Lehnert, W. (1992) "Classifying Texts Using Relevancy Signatures" to appear in *Proceedings of the Tenth National Conference on Artificial Intelligence.* San Jose, CA.